E1 277 Reinforcement Learning

Assignment 1 – Programming

January 2022

1 The Problem

Assume that our mess on the IISc campus wants to open a cheese station similar to the egg counter. The cheese must be purchased at the counter of this station. The following contains the details in this scenario.

- Every weekday morning, the mess master must decide how much cheese to buy from the vendor.
- Each cheese packet has $100, 200, 300, 400$ or 500 slices.
- Every workday, the mess committee can purchase only one packet of cheese or nothing at all.
- Each cheese slice is estimated to cost 10 rupees.
- Each cheese slice costs 12 rupees to purchase from the mess counter.
- During weekdays, the mess fridge can only hold 500 pieces of cheese slices.
- However, owing to the weekend cleaning routine, the fridge does not operate on weekends, thus there is no cheese counter on Saturdays and Sundays.
- Any cheese slices that haven't been sold by Friday evening will be wasted as it will not be consumable.
- If the number of slices purchased and the number of existing slices exceeds the capacity on a weekday, the extra inventory is wasted.
- The distribution of demand on any given day is given as follows: For 100, 200, 300, 400, and 500 cheese slices, respectively, the corresponding probabilities are $0.15, 0.05, 0.3, 0.25, 0.25$.
- Because the mess committee cannot sell more cheese than the amount of slices available, sales in a day are limited by the minimum of demand and available slices.

Now giving the context, we need to maximize the total expected profit of the mess over the whole week. The environment is given by the *IIScMess* class. In this class the demand at each of the days is given by *demand_value* and the corresponding probabilities are given by *demand_probs*. On each weekday, the mess master can buy cheese from the following set $\{0, 100, 200, 300, 400, 500\}$ of cheese slices. So, the possible action spaces are given by the *action_space* variable. The state is a tuple of the day of the week (or the weekend) and the starting cheese level for that day, for example ("Tuesday", 100). Next, *get_next_state_reward* is a method that calculates the next state and the reward along with the current state, the action, and the demand. The template also has another method called *get_transition_prob* that returns all possible transitions and rewards for a given state-action pair using the *get_next_state_reward* method, together with the corresponding probabilities.

Your task is to complete the functions *iterative_policy_evaluation* and *value_iteration* using the algorithm taught in the class. One can refer the algorithms from chapter 4 of Sutton Barto textbook. The *example_policy* function is demo policy. Assume all the policies are deterministic.

A policy is given in the form of a dictionary that maps states to action probabilities, for example, Policy = {('Monday', 0): {100: 0.4, 300: 0.6}, ('Tuesday', 0): {100: 0.4, 300: 0.6}...}. And the value of states is given by a dictionary such as Values v = {('Monday', 0): 2884.0, ('Tuesday', 0): 2204.0, ('Tuesday', 100): 3204.0, ...}.

2 Instructions for submission

We use GradeScope for evaluating this programming assignment. The students are provided with the solution template file. There are some functions which are to be coded by them. Once the coding is done, upload the solution file in Gradescope. This assignment is for 5 marks, where the test-cases for 3 marks are visible. The remaining 2 marks are from the hidden test cases.

2.1 Creating an account in GradeScope and submitting assignment

1. Go to https://www.gradescope.com/ and create a new account using the IISc email ID.
2. Join the RL course using the code: **ERK3EZ**
3. Go to the Assignments tab. Click on "Programming Assignment 1" and you should be able to submit your solutions there.

2.2 Important points

Below are the important points to keep in mind during this assignment submission.

1. Just fill in the code for the missing functions in the template.
2. DO NOT change the file name / signature of any class or functions.
3. Try not to hard-code any variable values. Use the variables in all the places. The hidden test cases can be of different environment and your code will fail if any hard-codings are present.
4. We have very strong plagiarism checks in place. Any malpractice would lead to severe consequences as per the institute policy.